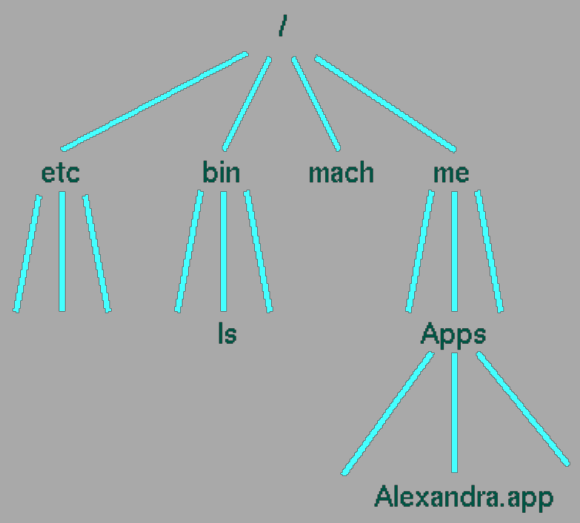
Les bases d'unix

Voici une petite introduction aux bases d'**unix** (les fichiers, les utilisateurs, les droits, le shell...). Cette présentation est spécifique à **NeXTSTEP** (qui utilise mach/BSD 4.3) mais est en fait assez générale et s'applique, dans son principe, à tous les systèmes unix.

- Les fichiers
- Les utilisateurs
- Les droits
- Le shell

Les fichiers

L'organisation des fichiers unix est hiérarchisée. À la base de l'arborescence, il y a la racine (root), symbolisé par le caractère "/". Les différentes branches de l'arborescence sont des répertoires. Pour représenter le nom d'un fichier dans une arborescence, on cite à partir de la racine le nom des répertoires séparés par le caractère "/". Ainsi "/me/Apps/Alexandra.app" est un nom complet du programme **Alexandra.app** :



Tous les caractères sont acceptés dans les noms des fichiers sauf le "/" et le caractère de code 0. Sous **NeXTSTEP** la limitation des noms de fichiers est de 255 caractères.

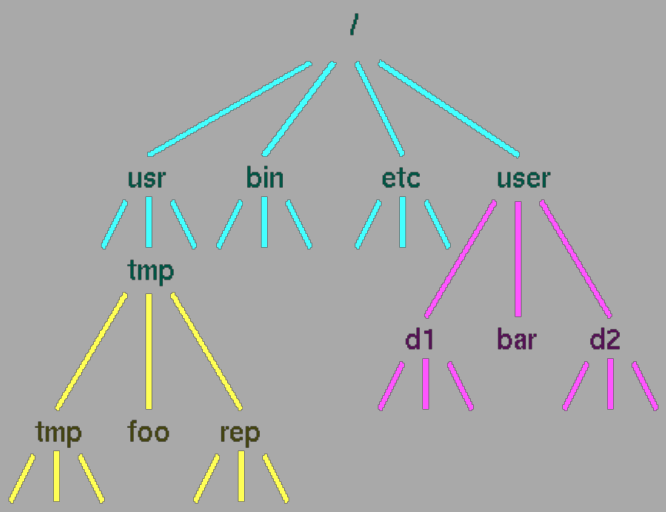
Sous unix tout est un fichier : une image GIF est un fichier,un répertoire est un fichier, le disque est un fichier, la mémoire est un fichier... Pour pouvoir représenter tout cela, il existe plusieurs type de fichiers :

- *Les fichiers "normaux"* : ce sont les fichiers sur disque, au sens classique du terme.
- *Les répertoires* : ce sont de petits fichiers qui contiennent une liste de nom d'autres fichiers. Le répertoire "." est le répertoire courant, et le répertoire ".." est le répertoire père (celui du dessus dans l'arborescence).
- *Les fichiers "spéciaux"* : ce sont des fichiers qui contiennent un numéro de driver. Par l'utilisation de ces fichiers, on accède directement à des drivers. Il existe ainsi des fichiers spéciaux pour tous les types de péripherique (port série, disque, mémoire...).
- *Les liens symboliques* : ce sont de petits fichiers qui contiennent le nom d'un fichier cible. Lorsqu'une application ouvre un lien symbolique, le système va en fait ouvrir le fichier cible de façon transparente.

Il existe d'autres type de fichiers un peu plus exotiques (comme les tubes).

Sous **NeXTSTEP**, tous les fichiers et les répertoires **unix** purs sont cachés à l'utilisateur lambda. On voit ces répertoires seulement si l'on coche "Expert Unix" dans le panneau de préférence suivant :

Une machine peut posséder plusieurs disques pouvant contenir eux-même plusieurs partitions. Chaque partition possède sa propre arborescence. Pour pouvoir accéder à ces partitions, le système "monte" ces partitions sur la partition root (celle qui contient le système).



Dans le dessin précédant, on a 3 partitions, dont 2 sont montées sur la partition root. **NeXTSTEP** monte les partitions des disques amovibles (CD-ROM, disquettes...)au premier niveau de la partition root en utilisant comme nom du point de montage (le "/" si la partition était seule), le nom du volume.

Sous unix, un fichier sur disque est caractérisé par une structure appelée "inode". Cette structure contient des dates, des tailles, des droits, les adresses des blocs sur disque... Le nom d'un fichier qui se trouve dans un répertoire est en fait un pointeur sur l'inode du fichier. Si "toto" est un fichier, il est possible de lui créer un lien, par exemple "tata". Et ainsi "toto" et "tata" sont 2 noms de fichiers pointant vers le même inode et donc vers un fichier unique sur disque. Si l'on modifie "toto", "tata" est aussi modifié vu que c'est le même fichier. L'inconvénient des liens "durs" est qu'ils ne marchent que dans une même partition, mais leur avantage est que si l'on déplace un des fichiers, le lien sera conservé car la référence de l'inode est inchangée.

Par opposition à ces liens appelés "durs" existent des liens symboliques. Si "toto" est un fichier sur disque, on peut créer un lien symbolique appelé "tata" qui pointera sur "toto". L'avantage des liens symboliques sur les liens "durs" est qu'ils marchent même entre les partitions, et même si le type de système de fichiers est différent. L'inconvénient est que si "toto" est déplacé, "tata" pointera vers rien du tout et le lien sera ainsi cassé.

Les utilisateurs

Unix est un système multiutilisateur. cela veut dire que plusieurs utilisateurs peuvent utiliser la machine, l'un après l'autre ou tous ensemble.

Quand plusieurs utilisateurs, à travers des consoles distantes, se connectent à une machine unix, chaque utilisateur a son propre environnement de travail (en particulier un répertoire maison appelé "home"), et peut se croire seul sur la machine.

Sous **NeXTSTEP** un utilisateur d'une machine peut se connecter à une autre machine pour travailler. Il peut utiliser des

ressources de cette machine distante comme les disques, les imprimantes, les modems... Mais l'utilisateur peut aussi lancer une application qui tournera sur la machine distante mais qui utilisera le clavier/écran de la machine locale. C'est le NXhosting possible grâce à Display PostScript. Les autres unix utilisent le **X Window System** développé par le MIT (la première version du système a été **W**, et nous en sommes aujourd'hui à la version 11 de **X**).

Chaque utilisateur doit se déclarer au système par un nom et un mot de passe, c'est ce que l'on appelle le login :

Pour maintenir l'intégrité du système, sont associés aux utilisateurs des droits d'accés aux ressources de la machine. Ces droits étant limités, les utilisateurs ne peuvent perturber un système complet. Il existe pourtant un utilisateur particulier appelé le *super utilisateur* ou **root**. Ce dernier a tous les droits sur la machine et peut tout faire.

Lorsque l'on travaille sur une machine unix on le fait en étant simple utilisateur (sous **NeXTSTEP** il existe un utilisateur créé par défaut et s'appelant *"me"*). On se connecte en **root** uniquement pour effectuer un travail administratif (ajout d'un nouvel utilisateur, changement de configuration matérielle...), mais on y reste le moins longtemps possible.

Normalement lorsque l'on installe un logiciel applicatif sous **NeXTSTEP**, on reste simple utilisateur et on place ce logiciel dans son répertoire applicatif ("/me/Apps" par exemple). Si l'on désire que d'autres utilisateurs de la machine puissent utiliser ce logiciel, il faut alors le placer dans un répertoire plus global comme /NextApps, mais là, pour des raisons de sécurité il faut faire l'installation en étant super utilisateur. Ceci peut se faire en terminant la session courante et en se reconnectant sous **root**. Sous **NeXTSTEP** on peut aussi utiliser OpenSesame pour lancer l'installation, ce qui permet d'être **root** pour un programme en restant connecté comme simple utilisateur (le mot de passe de root est demandé).

Les droits

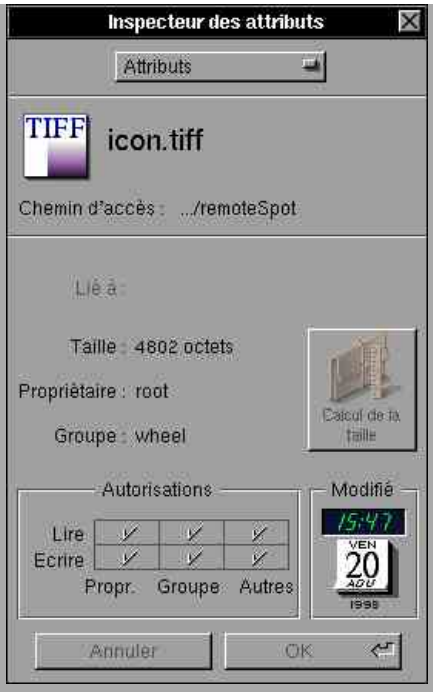
Chaque utilisateur est enregistré dans le système par son nom de login. Une correspondance est faite entre ce nom de login et un numéro appelé User Identification (uid). Ceci estréalisé grâce à la base de données NetInfo. Le uid valant 0 représante le super utilisateur **root**.

Un utilisateur fait parti d'un groupe d'utilisateur, groupe défini lors de la création de l'utilisateur par l'administrateur du système à l'aide du programme User Manager. Au groupe est associé un numéro appelé Group Identification (gid).

Chaque utilisateur est ainsi, dès son login, associé à un couple uid/gid.

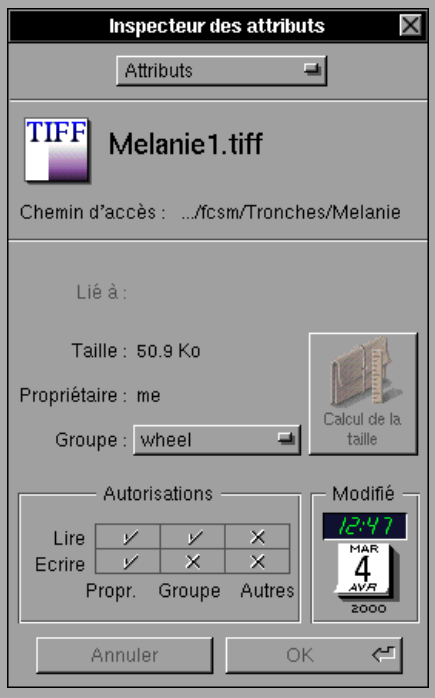
Tous les fichiers du système appartiennent à quelqu'un. Ainsi, à tous les fichiers est associé un couple utilisateur/groupe (uid/gip) représentant le propriétaire du fichier, c'est-à-dire celui qui a créé le fichier. Par exemple quand l'utilisateur "me/wheel" crée un nouveau fichier "nom.gif", ce fichier appartient à "me/wheel".

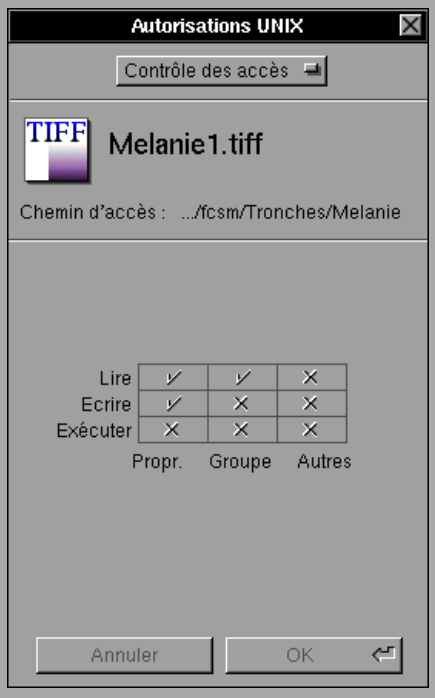
Dans l'exemple suivant utilisant un inspecteur du WorkSpace Manager, on voit que le fichier "icon.tiff" appartient à root/wheel :



Chaque fichier possède des autorisations d'accès. Il y a des autorisations pour le propriétaire du fichier, pour les personnes du groupe du fichier, et pour toutes les autres.

Pour chaque type d'autorisation, on a des droits en lecture, en écriture, et en exécution. Dans l'exemple suivant, le fichier, qui appartient à "me/wheel", peut être lu et écrit par "me/staff" (car "me" est le propriétaire), il peut être juste lu par les personnes du groupe "wheel" (comme "you/wheel"), et il est interdit d'accès pour tous les autres (comme "other/staff").





Il existe pourtant une exception à ces autorisation : il s'agit de **root** qui a le droit de tout faire quelque soient les droits attribués et quelque soit le propriétaire. Ainsi, dans l'exemple précédant, le fichier est accessible à **root**.

Il existe un masque des droits utilisés lors de la création d'un fichier par un utilisateur. Ce masque indique les types de droit qui peuvent être utilisés. Par exemple, avec les droits définis dans le panneau suivant, les fichiers créés par l'utilisateur n'auront pas les droits d'écriture pour le groupe, et aucun droit pour tous les autres utilisateurs.

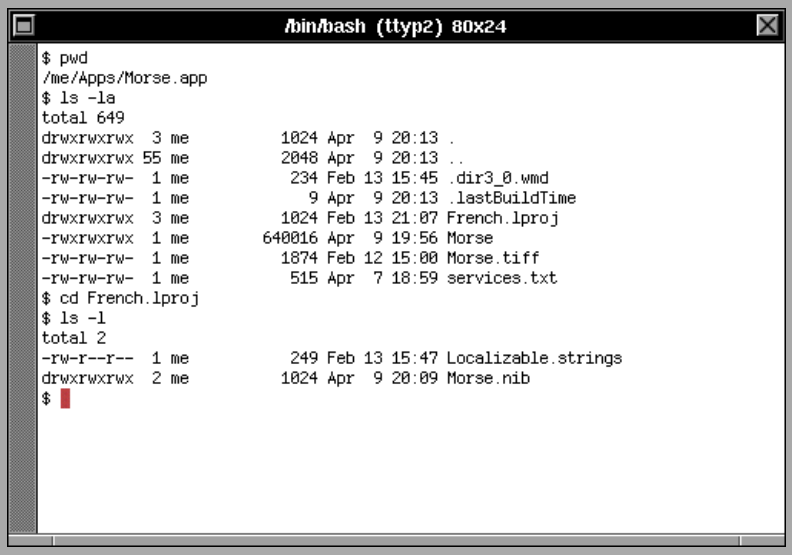
Le shell

Le shell est un programme qui permet d'interagir avec le système **unix** à l'aide de lignes de commande. C'est ce que l'on appelle aussi le CLI (CommandLine Interpreter), par opposition au GUI (Graphic UserInterface) des systèmes graphiques comme **NeXTSTEP**.

Il existe en fait un vaste choix de shells ayant des fonctionnalités et des syntaxes diverses. Les deux shells de base (historiquement parlant) sont **sh** le Bourne Shell, et **csh** le C Shell. Les exemples qui suivent utilisent un dérivé de **sh**.

Avec **NeXTSTEP** existe un programme s'appelant Terminal servant à émuler une console. On peut lancer avec **Terminal** plusieurs fenêtres à la fois, et ont peut choisir son shell préféré.

L'exemple suivant montre quelques commandes unix tapées avec un shell :



Il existe des centaines, voire des milliers de commandes disponibles sous unix. Certaines commandes sont sont internes au shell, d'autres sont des programmes externes. Le signe "$" représente le "prompt" et indique que le shell attend une commande.

Les commandes ont généralement des noms assez court et peuvent avoir des options de lancement. Ces options sont précédées par le signe "-". Par exemple la commande **ls** permet de lister les fichiers d'un répertoire, avec l'option "-l", le listing a un format long (on a un descriptif pour chaque fichier); avec l'option "-a" s'affiche aussi les fichiers normalement cachés commençant par un point. L'option "-la" est équivalent à "-l -a".

Le shell utilise certains caractères spéciaux comme le "*" qui indique n'importe quel caractère. Ainsi "*.txt" représente tous les fichiers se terminant pas ".txt"; et "a*b" représente tous les fichiers commençant par "a" et finissant par "b". Les caractères "[]" indiquent une plage de caractères. Ainsi "[a-k]*" représente tous les fichiers commançant par une lettre entre "a" et "k". Le caractère "?" indique n'importe quel caractère. Ainsi "a?b" représente un nom de fichier de 3 caractères, le deuxième étant quelconque (alors que "a*b" représente des noms qui peuvent avoir plus de 3 caractères).

Commande	Description	Exemple

ls	Affichage de la liste des fichiers du répertoire courant ou du répertoire donné en paramètre. L'option "-l" donne un format long (beaucoup d'informations sur les fichiers). L'option "-a" affiche les fichiers cachés.	ls -l /bin
cat	Affichage du contenu du fichier donné en argument.	cat fic1
cd	Changement de répertoire.	cd /usr/bin cd ..
pwd	Affichage du nom du répertoire courant.	pwd
mkdir	Création d'un nouveau répertoire.	mkdir rep
cp	Copie de fichiers	cp fic1 fic2 cp *.txt rep
mv	Changement de nom d'un fichier, et déplacement de fichiers.	mv old new mv fic1 fic2 rep
ln	Création d'un lien "dur" sur un fichier. L'option "-s" permet de créer un lien symbolique.	ln old new
rm	Effacement de fichiers.	rm fic1
rmdir	Effacement de répertoires.	rmdir rep1
ps	Affichage la liste des programmes tournant. Les options "axl" donnent la liste de tous les programmes qui tournent dans un format long.	ps -axl
kill	Envoi d'un signal à un programme. Par défaut un signal demandant au programme de s'arrêter est envoyer. L'option "-9" force l'arrêt du programme.	kill 123 kill -9 123
grep	Recherche d'une chaîne de caractères dans des fichiers.	grep nom *.txt
more	Affichage page par page d'un fichier.	more fic1
wc	Comptage du nombre de caractères, de mots et de lignes d'un fichier. L'option "-l" affiche uniquement le nombre de lignes.	wc -l fic1
man	Affichage de l'aide d'une commande unix. L'option "-k" permet de recherche un mot dans le titre de toutes les commandes.	man commande man -k mot

Par défaut les commandes affichent leur résultat sur la console, mais on peut rediriger celui-ci vers un fichier :

```
cat fic1 fic2 > fic3
```

Le contenu des deux fichiers fic1 et fic2 sont concaténés dans le fichier fic3.

`cat fic4 >> fic3`	Le contenu du fichier fic4 est ajouté au fichier fic3.

On peut aussi envoyer le résultat d'une commande comme entrée d'une autre commande :

`grep nom *.txt | more`	Recherche de la chaîne "nom" dans tous les fichiers "*.txt" du répertoire courant et afficher le résultat page par page.
`grep nom fic1 | wc -l`	Affiche le nombre de lignes où la chaîne "nom" apparaît dans le fichier "fic1".

Sur le système **NeXTSTEP** existe des commandes permettant d'interagir avec l'interface graphique. On trouve par exemple les commandes "open", "openfile", "copy" et "paste".

`open index.html`	On simule un double-clique sur le fichier HTML ce qui lance le butineur par défaut (typiquement OmniWeb).
`grep root fic1 | openfile`	On recherche toutes les lignes de fic1 contenant le mot "root", et on affiche le résultat avec Edit.
`ps -axl | copy`	On met dans le presse-papier le résultat de la commande "ps". On pourra donc utiliser "Command-v" pour copier le texte dans n'importe quelle application.
`paste | grep toto`	On affiche toutes les lignes contenant "toto" se trouvant dans le presse papier.

Pour la petite histoire il existe un programme pour construire des shell scripts graphiquement : ScriptWriter.